Replication Materials for, "The Coalition-Directed Vote in Contexts with Multi-party Governments"

Raymond M. Duch
raymond.duch@nuffield.ox.ac.uk
Nuffield College
University of Oxford
OX1 1NF Oxford UK

Jeff May
jbmay@uh.edu
Department of Political Science
University of Houston
Houston, TX 77204-3011

David A. Armstrong II*
armstrod@uwm.edu
University of Wisconsin-Milwaukee
Dept of Political Science
PO Box 413
Milwaukee, WI 53201-0413

January 14, 2011

All of the code below can be run on **R** version 2.11.1. It can also be run on older versions as well (probably at least as far back as 2.8(ish)), but has been tested on 2.11.1 recently.

This replication works by having all the files in a structured directory. The replication directory has the following contents:

```
- country_files/
- input_files/
- out_files/
```

The first two directories contain all the files needed to run this replication. The country data files are in the directory `country_files`. The `input_files` folder contains the files:

- `apsr_input.csv`,
- `party_labels.csv`,
- `manifesto.csv`,
- `ms_data_apsr.csv`,
- `country_map.txt`,
- `study_map.txt`,
- `martin_stevenson_repdata.dta`,

*Please direct questions regarding replication to Dave Armstrong.

- `mnl2.cmd`,
- `mnl2.mod`,
- `mnl3.cmd`, and
- `mnl3.mod`.

The `out_files` folder is empty and will be the destination of all figures created and of output files created during the replication process.

1 Environment set-up

To set the environment for the replication two steps need to be followed.
First, the following libraries are needed to execute the replication:

```
library(lattice)
library(foreign)
library(survival)
library(lme4)
library(runjags)
library(coda)
library(foreign)
library(plotrix)
```

Second, the master directory were all files are located needs to be specified. In this case it is `J:/Duch/APSR_Replication`, but as long as it is specified it can be anything.

```
replication_directory <- "J:/Duch/APSR_Replication"

setwd(replication_directory) ### Set directory were replication material is located
```

2 Figure 1

The code below will produce Figure 1 from the article and save it as a pdf file in the folder `out_files`.

```
x <- .25
p <- c(-1,0,1)
m1 <- c(.5,.6,.7,.8,.9)
res <- list()
gam3 <- cbind(rep(1,101), 0, 0)
gam2 <- cbind(seq(0,1,length=101), seq(1,0,length=101), 0)
gam1 <- cbind(rep(.5, 101), rep(.5, 101), 0)

for(j in 1:5){
s1 <- matrix(c(
    1,0,0,
```

```
    m1[j],(1-m1[j]),0,
    .5, 0, .5), ncol=3, byrow=T)

s2 <- matrix(c(
    0,1,0,
    m1[j],(1-m1[j]),0,
    0,.5,.5), ncol=3, byrow=T)

s3 <- matrix(c(
    0,0,1,
    .5,0,.5,
    0,.5,.5), ncol=3, byrow=T)

z1 <- s1 %*% p
z2 <- s2 %*% p
z3 <- s3 %*% p

u1 <- 4 - gam1 %*% (x - z1) ^2
u2 <- 4 - gam2 %*% (x-z2)^2
u3 <- 4 - gam3 %*% (x-z3)^2
res[[j]] <- cbind(u1,u2,u3)
}
res2 <- NULL
for(i in 1:5){
    res2 <- rbind(res2, cbind(res[[i]], gam2[,1], m1[i]))
}
colnames(res2) <- c("u1", "u2", "u3", "gamma2", "share")
res2 <- as.data.frame(res2)

pdf("out_files/figure1.pdf", height=4, width=4)
trellis.par.set(strip.background=list(col="gray90"))
key.res <- list(space="bottom", text=list(c("U(Party 1)", "U(Party 2)",
     "U(Party 3)")), lines=list(lty=c(1,2,3)))
xyplot(u1 ~ gamma2 | as.factor(share), data=res2, as.table=T,
     ylim=c(min(res2$u1), max(res2$u2)), key = key.res, layout=c(2,3),
     xlab="Probability Party 2 is Single-Party Governor", ylab="Utility",
     panel =function(x,y,subscripts){
        panel.lines(x,y, lty=1, col="black")
        panel.lines(x,res2$u2[subscripts], lty=2, col="black")
        panel.lines(x, res2$u3[subscripts], lty=3, col="black")
})
dev.off()
```

3 Figure 2

To replicate Figure 2 the code below assumes that you have followed the replication directory structure and that you have set-up the environment (as described above). The resulting image will be saved as a pdf file in the folder `out_files`.

```
setwd(replication_directory) ### Set directory were replication material is located

input <- read.csv("input_files/apsr_input.csv")[,-1]

dats <- as.character(input[,1])
lrs <- as.character(input[,11])
pvar <- as.character(input[,10])

# Generate means by country-party-election
i <- 1
tmp <- read.dta(paste("country_files/",dats[i],sep=""), convert.factors=F)
tmp[[lrs[i]]] <- scale(tmp[[lrs[i]]])
tmp.ag <- aggregate(tmp[[lrs[i]]], list(tmp[[paste(pvar[i], "_num", sep="")]]),
     mean, na.rm=T)
tmp.ag$cnumb <- input[i,3]
tmp.ag$year <- input[i,7]
tmp.ag$month <- input[i,6]
tmp.ag$studyno <- input[i,4]
agdata <- tmp.ag

for(i in 2:length(dats)){
   tmp <- read.dta(paste("country_files/",dats[i],sep=""), convert.factors=F)
    if(!is.null(tmp[[lrs[i]]])){
    tmp[[lrs[i]]] <- scale(tmp[[lrs[i]]])
    tmp.ag <- aggregate(tmp[[lrs[i]]], list(tmp[[paste(pvar[i], "_num", sep="")]]),
         mean, na.rm=T)
    tmp.ag$cnumb <- input[i,3]
    tmp.ag$year <- input[i,7]
    tmp.ag$month <- input[i,6]
    tmp.ag$studyno <- input[i,4]
    agdata <- rbind(agdata, tmp.ag)
    }
    cat(i, " ")
}

cmap <- read.table("input_files/country_map.txt", header=T)
agdata$country <- cmap[match(agdata$cnumb, cmap[,2]), 1]
# Read in and manage the manifesto data
man <- read.csv("input_files/manifesto.csv", header=T)
fun <- function(x){
    un <- unique(x)
```

```
    un <- un[order(un)]
    match(x, un)
}

man$enum <- c(unlist(by(man$isodate, list(man$cnumb), fun)))
man$cpe <- paste(man$cnumb, man$partynum, man$enum, sep=".")

plab <- read.csv("input_files/party_labels.csv")


agdata$cp <- paste(agdata$cnumb, agdata[,1], sep=".")
plab$cp <- paste(plab$cnumb, plab$Party.Numb, sep=".")
agdata$mancode <- plab[match(agdata$cp, plab$cp), "Manifesto.Code"]

elecs <- by(man[,c("year", "month", "day")], list(man$cnumb),
     function(y)ISOdate(y[,1], y[,2], y[,3]))
elecs <- lapply(elecs, unique)
agdata$isodate <- ISOdate(agdata$year, agdata$month, 1)

unagc <- unique(agdata[,3])
noin <- seq(1,length(elecs))[-which(names(elecs) %in% unagc)]
noin <- rev(noin)
for(i in 1:length(noin)){
    elecs[[noin[i]]] <- NULL
}

agdata$elecnum <- NA
for(i in 1:length(elecs)){
dists <- NULL
    for(j in 1:length(elecs[[i]])){
        dists <- cbind(dists, (agdata$isodate[
           which(agdata$cnumb == names(elecs)[i])] - elecs[[i]][j]))
    }
    dists <- apply(dists, 2, abs)
    (agdata$elecnum[which(agdata$cnumb == names(elecs)[i])] <-
        apply(dists, 1, which.min))
}

agdata <- agdata[-which(is.na(agdata$mancode)), ]
agdata$cpe <- paste(agdata$cnumb, agdata$mancode, agdata$elecnum, sep=".")
agdata$edate <- man$isodate[match(agdata$cpe, man$cpe)]

agdata <- agdata[order(agdata$cnumb, agdata$isodate, agdata[,1]), ]
agdata <- na.omit(agdata)

agdata$place2 <- man[match(agdata$cpe, man$cpe), "place2"]
```

```
# Note that as of this writing, the lme4 package is not available as a
# pre-compiled binary for Mac OSX.  Interested users will have to
# download the source code from cran and install that way.


names(agdata)[2] <- "mean_lr"
mod1b <- lmer(mean_lr ~ place2 + (1+place2|cnumb), data=agdata)

preds <- rep(NA, nrow(agdata))

for(i in 1:nrow(coef(mod1b)[[1]])){
    inds <- which(agdata$cnumb == rownames(coef(mod1b)[[1]])[i])
    td <- cbind(1, agdata[inds, "place2"])
    preds[inds] <- td %*% matrix(unlist(coef(mod1b)[[1]][i,]), ncol=1)
}

agdata$pred.places <- preds
write.csv(agdata, "out_files/pred_places_apsr_rep.csv")

pdf("out_files/figure2.pdf", height=4, width=4)
trellis.par.set(strip.background = list(col="gray90"))
print(
xyplot(mean_lr ~ preds | country, data=agdata, pch=16, cex=.4, col="black",
    as.table=T,  par.strip.text=list(cex=.7),
    xlab="Manifesto Placement", ylab="Mean Left-Right Self-Placement")
)
dev.off()
```

4 Martin and Stevenson Replication

This next section not only replicates the Martin and Stevenson (2001) model and generates Figure 8 in the Appendix but also generates predicted probabilities of coalition membership needed later in the replication and saved in `out_files/ms_data_apsr_rep.csv`.

```
setwd(replication_directory)
ms.data <- read.dta("input_files/martin_stevenson_repdata.dta")

mod <- coxph(Surv(time, realg) ~ minor + minwin + numpar + dompar +
     medparty + govdiv + minorxmgd + prevpm + antisystem + statusquo +
     minorxinv + pact + antipact + strata(strat),
     data=ms.data, method="exact")

pred.prob <- predict(mod, type="exp")
ms.data$pred.prob <- pred.prob
```

```
write.csv(ms.data, "out_files/ms_data_apsr_rep.csv")

ms.data$elec <- as.Date(ms.data$elec, origin="1960-1-1")
ms.data$cntry_elec <- paste(ms.data$country, ms.data$elec, sep=".")
top_3_probs <- by(cbind(ms.data$realg, pred.prob), list(ms.data$cntry_elec),
     function(x)x[which(rank(1/x[,2]) %in% 1:3), ])
top_3_probs <- do.call(rbind, top_3_probs)
t3p_names <- strsplit(rownames(top_3_probs), split=".", fixed=T)
t3p_names <- t(sapply(t3p_names, function(x)x[1:2]))
top_3_probs$country <- t3p_names[,1]
top_3_probs$date <- as.Date(t3p_names[,2], "%Y-%m-%d")

pdf("out_files/figure_8.pdf", height=4, width=4)
trellis.par.set(strip.background=list(col="gray90"))

xyplot(pred.prob ~ date | country, data=top_3_probs, xlab = "Time",
    ylab = "Pr(Coalition Formation)", as.table=T,
    scales=list(x=list(rot=45)), par.strip.text=list(cex=.6),
    panel = function(x,y,subscripts){
       panel.points(x[which(top_3_probs$V1[subscripts] == 0)],
          y[which(top_3_probs$V1[subscripts] == 0)],pch=16,
          cex=1, col="gray75")
       panel.points(x[which(top_3_probs$V1[subscripts] == 1)],
          y[which(top_3_probs$V1[subscripts] == 1)],pch=16,
          cex=1, col="black")
    })
dev.off()
```

5 Netherlands Example

```
setwd(replication_directory)

ms.data <- read.csv("input_files/ms_data_apsr.csv")
neth.coal <- ms.data[which(ms.data$country == "The Netherlands"),]
un_ne <- unique(neth.coal$elec)
places <- read.csv("out_files/pred_places_apsr_rep.csv")
places$edate <- as.Date(as.character(places$edate), "%Y-%m-%d")
neth.places <- places[which(places$edate == as.Date(as.character(un_ne[3]),
    "%Y-%m-%d"))[4:7], ]
neth.coal <- neth.coal[which(neth.coal$elec == un_ne[3]), ]
neth_ind <- which(ms.data$elec == un_ne[3])
neth_ind <- neth_ind[apply(neth.coal[, grep("coal", colnames(neth.coal))], 1,
    function(x)!(22310 %in% x))]
neth.probs <- ms.data$pred.prob[neth_ind]

neth.pnum <- c(22320, 22420, 22330, 22521)
```

```
dma.num <- c(20,70,22,63)
names(neth.pnum) <- names(dma.num) <- c("PvdA","VVD","D66","CDA")



neth_coals <- neth.coal[, grep("coal", colnames(neth.coal), fixed=T)]
neth_coals <- neth_coals[apply(neth_coals, 1, function(x)!(22310 %in% x)), ]
neth_coals <- neth_coals[,1:4]
neth_coals_party <- apply(neth_coals, 2,
    function(x)names(neth.pnum)[match(x, neth.pnum)])

rmna <- function(x){
    if(sum(is.na(x)) > 0){
        tmp.x <- x[-which(is.na(x))]
    }
    else{
        tmp.x <- x
    }
    tmp.x
}
nc.str <- apply(neth_coals_party, 1, function(x)paste(rmna(x), collapse = "-"))

pvda.ind <- grep("PvdA", nc.str)
vvd.ind <- grep("VVD", nc.str)
d66.ind <- grep("D66", nc.str)
cda.ind <- grep("CDA", nc.str)

neth.gams <- matrix(nrow=length(nc.str), ncol=4)
neth.gams[pvda.ind,1] <- round(neth.probs[pvda.ind], 4)
neth.gams[d66.ind,2] <- round(neth.probs[d66.ind], 4)
neth.gams[vvd.ind,3] <- round(neth.probs[vvd.ind], 4)
neth.gams[cda.ind,4] <- round(neth.probs[cda.ind], 4)
n.neth.gams <- apply(neth.gams, 2, function(x)x/sum(x, na.rm=T))
n.neth.gams <- round(n.neth.gams, 3)
gam.tab <- cbind(nc.str, neth.gams, n.neth.gams)
gam.tab[which(is.na(gam.tab))] <- ""

n.neth.gams[which(is.na(n.neth.gams), arr.ind=T)] <- 0


n86.dat <- read.dta("country_files/neth86ds.dta")
neth.tab <- table(n86.dat[which(n86.dat$vote_num %in% dma.num), "vote_num"])
names(neth.tab) <- neth.pnum[match(names(neth.tab), dma.num)]
neth.seats <- apply(neth_coals, 2, function(x)neth.tab[match(x, names(neth.tab))])
neth.seats[which(is.na(neth.seats), arr.ind=T)] <- 0
neth.seats <- prop.table(neth.seats, 1)
```

```
neth.parties <- apply(neth_coals, 2,
    function(x)neth.places$pred.places[match(x, neth.places$mancode)])
neth.parties[which(is.na(neth.parties), arr.ind=T)] <- 0

Z <- diag(neth.parties %*% t(neth.seats))

x <- -2

pvda.dist <- sum(((x-Z[pvda.ind])^2)*n.neth.gams[pvda.ind, 1])
d66.dist <- sum(((x-Z[d66.ind])^2)*n.neth.gams[d66.ind, 2])
vvd.dist <- sum(((x-Z[vvd.ind])^2)*n.neth.gams[vvd.ind, 3])
cda.dist <- sum(((x-Z[cda.ind])^2)*n.neth.gams[cda.ind, 4])

new.pvda <- x + sqrt(pvda.dist)
new.d66 <- x + sqrt(d66.dist)
new.cda <- x + sqrt(cda.dist)
new.vvd <- x + sqrt(vvd.dist)

plmat <- cbind(neth.places$pred.places, c(new.pvda, new.d66, new.cda, new.vvd))
rownames(plmat) <- neth.places$mancode


xmin <- min(c(plmat))
xmax <- max(c(plmat))
prg <- c(xmin, xmax)
sig <- c(-1,1,-1,1)
pdf("out_files/figure_3.pdf", height=4, width=4)
plot(c(xmin, xmax), c(-.2,3.2), type="n", axes=F, xlab="", ylab="")
text(mean(c(xmin, xmax)), 3, "PvdA", cex=5, col="gray70")
text(mean(c(xmin, xmax)), 2, "D66", cex=5, col="gray70")
text(mean(c(xmin, xmax)), 1, "CDA", cex=5, col="gray70")
text(mean(c(xmin, xmax)), 0, "VVD", cex=5, col="gray70")
arrows(rep(xmin, 4), 0:3, rep(xmax,4), 0:3, angle=90, code=3, length=0.1)
points(plmat[,1], 3:0, pch=16, cex=1.5, col="black")
points(plmat[,2], 3:0, pch=15, cex=1.5, col="red")
arrows(plmat[,1], seq(3.025, .025, by=-1), plmat[,2] + .01*sig,
    seq(3.025, .025, by=-1), length=.1)
dev.off()
```

6 Bayesian Estimation

Here we discuss the considerable data preparation required to get the data into the correct form. It is worth noting here that the predictions of party placements (from the previous section) were originally made with parties from over 300 studies. At that time, we had planned on using all of the studies, though many of them showed no signs of convergence, so we used

the 86 studies that showed signs of convergence. As these rescaled party placements are based on a multi-level linear model, the predictions change somewhat when we truncate and make the predictions based on only the datasets used in the article. These two sets of predictions correlate at roughly 0.98 with the biggest departures coming at the extremes of the scale (i.e., where it is likely to matter least). For the sake of completeness we also include these original predictions in a file called `original_party_placements.csv`.

6.1 Merge and Prepare Data

```
setwd(replication_directory)

ms.data <- read.csv("input_files/ms_data_apsr.csv")[,-1]
ms.data$cnumb <- as.numeric(do.call(rbind, strsplit(as.character(ms.data$strat),
    split=".", fixed=T))[,1])
ms.data$ced <- paste(ms.data$cnumb, ms.data$elec, sep=".")
strats <- unlist(by(as.character(ms.data$strat), list(ms.data$ced),
    function(x)unique(as.character(x))))

pred.places <- read.csv("out_files/pred_places_apsr_rep.csv")[,-1]
pred.places$cmy <- paste(pred.places$cnumb, pred.places$month,
    pred.places$year, sep=".")
pred.places$edate <- as.Date(as.character(pred.places$edate), "%Y-%m-%d")
pred.places$ced <- paste(pred.places$cnumb, pred.places$edate, sep=".")
pred.places$strat <- strats[match(pred.places$ced, names(strats))]

plabs <- read.csv("input_files/party_labels.csv")[,-1]
plabs$cp <- paste(plabs$cnumb, plabs$Party.Numb, sep=".")
source("input_files/study_map.R")
input <- read.csv("input_files/apsr_input.csv")[,-1]
controls <- apply(input[,12:ncol(input)], 2, as.character)

zzindat <- match(map.mat[,2], input$studynum)

for(ii in 1:86){
    dnum <- zzindat[ii]
    datname <- as.character(input[dnum,1])
    dat <- read.dta(paste("country_files/",datname,sep=""), convert.factors=F)

    depvar <- as.character(input[dnum,10])
    depvar <- paste(depvar, "num", sep="_")

    dat$cp <- paste(as.numeric(input[dnum,3]), dat[[depvar]], sep=".")
    dat$cp[which(is.na(dat[[depvar]]))] <- NA
    dat$manparty <- plabs[match(dat$cp, plabs$cp), 4]

    unmanparty <- na.omit(unique(dat$manparty))
```

```
unmanparty <- unmanparty[order(unmanparty)]

dat$v <- match(dat$manparty, unmanparty)

lr <- dat[[as.character(input[dnum,11])]]
lr <- (lr-mean(lr, na.rm=TRUE))/sd(lr, na.rm=TRUE)

ctrl_names <- controls[dnum, ]
ctrl_names <- ctrl_names[-which(is.na(ctrl_names))]
ctrl_names <- ctrl_names[which(ctrl_names %in% names(dat))]
ctrl_names <- as.vector(c("age", ctrl_names[-which(ctrl_names == "age")]))

ctrl_mat <- dat[,match(ctrl_names, names(dat))]
ctrl.na <- apply(ctrl_mat, 2, function(x)mean(is.na(x)))
if(max(ctrl.na) == 1){
    ctrl_mat <- ctrl_mat[,-which(ctrl.na == 1)]
}

ctrl_mat <- cbind(ctrl_mat, 1)
tabs <- apply(ctrl_mat[,2:ncol(ctrl_mat)], 2, table)
maxtabs <- max(sapply(tabs, length))
tabs <- lapply(tabs, function(x)x/sum(x))
tabs <- t(sapply(tabs, function(x)c(x, rep(0, maxtabs-length(x)))))
colnames(tabs) <- rownames(tabs) <- NULL
prob.mat <- rbind(0, tabs)

p.vote <- table(dat[["v"]])/sum(table(dat[["v"]]))

age.mean <- mean(ctrl_mat[,1])
age.tau <-1/var(ctrl_mat[,1])

input.cmy <- paste(input[dnum,3], input[dnum,6], input[dnum,7], sep=".")
places <- pred.places[which(pred.places$cmy == input.cmy), ]

coals <- ms.data[which(ms.data$strat == places$strat[1]), 16:29]
ins <- which(apply(coals, 1, function(x)sum(!(x %in% c(unmanparty, NA)))) == 0)
coals <- coals[ins, ]

place_mat <- sapply(1:14, function(x)places[match(coals[,x], places$mancode),
     "pred.places"])
place_mat[which(is.na(place_mat), arr.ind=TRUE)] <- 0

pred.seats <- sapply(1:length(unmanparty), function(x)sum(dat$v == x, na.rm=T))
seats_mat <- sapply(1:14, function(x)pred.seats[match(coals[,x], unmanparty)])
seats_mat[which(is.na(seats_mat), arr.ind=TRUE)] <- 0
seats_mat <- prop.table(seats_mat, 1)
```

```
    coal.probs <- ms.data$pred.prob[which(ms.data$strat == places$strat[1])]

    strat.places <- gamma <- NULL
    code <- 0
    for(i in 1:length(unmanparty)){
        ins2 <- which(apply(coals, 1, function(x)max(unmanparty[i] %in% x)) == 1)
        if(i > 1){
            if(length(ins2) != li2p){
                code <- 1
            }
        }

        strat.places <- cbind(strat.places,
        diag(place_mat[ins2, ] %*% t(seats_mat[ins2, ])))
        gamma <- cbind(gamma, coal.probs[ins2])

        li2p <- length(ins2)

        if(code == 0){
            sin.places <- places[match(unmanparty, places$mancode), "pred.places"]
        }
    }

    gamma <- prop.table(gamma, 2)
    plmat <- cbind(sin.places, 0)
    mins <- apply(ctrl_mat, 2, min, na.rm=T)
    for(i in 2:ncol(ctrl_mat)){
        ctrl_mat[,i] <- ctrl_mat[,i] + (1-mins[i])
    }
    N <- nrow(dat)

    Nctrl <- ncol(ctrl_mat)
    Np <- length(unmanparty)
    vote <- dat$v
    ctrlmat <- as.matrix(ctrl_mat)
    dimnames(ctrlmat) <- dimnames(vote) <- dimnames(plmat) <- NULL
    prob.mat <- as.matrix(prob.mat)
    dimnames(prob.mat) <- NULL
    gwd <- getwd()

if(!file.exists("jags1")){ dir.create("jags1")}

    lf <- list.files("jags1")
    if(!(as.character(ii) %in% lf)){
        dir.create(paste("jags1/",as.character(ii),sep=""))
```

```
    }
    setwd(paste("jags1/",as.character(ii),sep=""))
    cat(dump.format(list(Nctrl=Nctrl, Np=Np, N=N, prob.mat=prob.mat,
        plmat = plmat, lr=lr, vote=vote, ctrlmat=ctrlmat, nc = nrow(gamma),
        gamma=gamma, z=strat.places)), file = "mnldat.txt", append=F)

    setwd(gwd)

if(!file.exists("jags2")){ dir.create("jags2")}

lf <- list.files("jags2")
    if(!(as.character(ii) %in% lf)){
        dir.create(paste("jags2/",as.character(ii),sep=""))
    }
    setwd(paste("jags2/",as.character(ii),sep=""))
    cat(dump.format(list(Nctrl=Nctrl, Np=Np, N=N, prob.mat=prob.mat,
        plmat = plmat, lr=lr, vote=vote, ctrlmat=ctrlmat, nc = nrow(gamma),
        gamma=gamma, z=strat.places)), file = "mnldat.txt", append=F)

    setwd(gwd)

    ## Generate Initial Values

    Np <- length(unmanparty)
    unlr <- unique(lr)
    unlr <- unlr[-which(is.na(unlr))]
    lr.start <- is.na(lr)
    lr.start[which(lr.start == F)] <- NA
    lr.start[which(lr.start == "TRUE")] <- sample(unlr,
         sum(lr.start == "TRUE", na.rm=T), replace=TRUE)

    vote.start <- is.na(dat$v)
    vote.start[which(vote.start == F)] <- NA
    vote.start[which(vote.start == "TRUE")] <- sample(1:length(unmanparty),
        sum(vote.start == "TRUE", na.rm=T), replace=TRUE)

    ctrl_mat_start <- apply(ctrl_mat, 2, function(x)is.na(x))
    ctrl_starts <- lapply(1:ncol(ctrl_mat), function(x)unique(ctrl_mat[,x]))
    for(i in 1:length(ctrl_starts)){
        sums <- sum(is.na(ctrl_starts[[i]]))
        if(sums > 0){
            ctrl_starts[[i]] <- ctrl_starts[[i]][-which(is.na(ctrl_starts[[i]]))]
        }
    }

    missings <- apply(ctrl_mat_start, 2, function(x) sum(x == "TRUE"))
```

```
    #ctrl_mat_start[which(ctrl_mat_start == FALSE, arr.ind=T)] <- NA
    for(i in 1:ncol(ctrl_mat_start)){
        ctrl_mat_start[which(ctrl_mat_start[,i] == FALSE), i] <- NA
    }

    for(i in 1:ncol(ctrl_mat_start)){
        if(missings[i] > 0){
            ctrl_mat_start[which(ctrl_mat_start[,i] == "TRUE"), i] <- sample(
            ctrl_starts[[i]], missings[i], replace=TRUE)
        }
    }

    lambda <- -1
    beta <- round(runif(N,0,1), 0)
    prob.beta <- .5
    phi <- matrix(NA, nrow=Np, ncol=ncol(ctrl_mat))
    phi[1:(nrow(phi)-1), ] <- runif((Np-1)*ncol(ctrl_mat), -.5,.5)


    vote <- vote.start
    lr <- lr.start
    ctrlmat <- as.matrix(ctrl_mat_start)
    dimnames(ctrlmat) <- NULL

    setwd(paste("jags1/",as.character(ii),sep=""))
    cat(dump.format(list(vote=vote, lr=lr, ctrlmat = ctrlmat, lambda=lambda,
        beta=beta, prob.beta=prob.beta, phi=phi, T=T)),
        file = "mnlinits.txt", append=F)
    setwd(gwd)

setwd(paste("jags2/",as.character(ii),sep=""))
    cat(dump.format(list(vote=vote, lr=lr, ctrlmat = ctrlmat, lambda=lambda,
        beta=beta, prob.beta=prob.beta, phi=phi, T=T)),
        file = "mnlinits.txt", append=F)
    setwd(gwd)

    cat(ii, " ")
}

setwd(replication_directory)

for(i in 1:86){
file.copy("input_files/mnl2.mod", paste("jags1/", i, "/mnl2.mod", sep=""))
    file.copy("input_files/mnl2.cmd", paste("jags1/", i, "/mnl2.cmd", sep=""))
file.copy("input_files/mnl3.mod", paste("jags2/", i, "/mnl3.mod", sep=""))
    file.copy("input_files/mnl3.cmd", paste("jags2/", i, "/mnl3.cmd", sep=""))
```

```
}
```

6.2 Running the Bayesian Models

The commands above require the use of files generated previously in the code, though we also provide them in the replication materials for those who want to start here. These models were run on JAGS version 1.0.3 on the high performance computer cluster at the University of Wisconsin–Milwaukee.[1] Running the models one at a time on a desktop computer could result in quite a long computation time as the mean run-time for each model is around 20 hours (resulting in around 78 days of compute-time on similar hardware).[2]

Two folders are created in the process above `jags1` and `jags2`. The first contains all the folders and files for the main analysis, while the second contains the folders and files for the sensitivity analysis described further down in this document. After installing JAGS, when in the numbered directory for a particular study (i.e. `jags1\75`, the folder for study # 75), if you are on a Mac, you can do all of the required computations by typing:

```
jags mnl2.cmd
```

In the Windows environment (tested on XP, SP3), first install JAGS. Then open up a command window (Start → Run and then type `cmd` in the box labeled "open") and change the directory to the one containing the data and model you wish to run. Then, type `jags mnl2.cmd`, just as above, and the model should run as expected.

The computation results in three files - two chain files labeled `CODAchain1.txt` and `CODAchain2.txt` as well as an index file `CODAindex.txt` which provides information about the parameters to which the values in the chain files refer. Diagnostics can then be done on these markov chains to evaluate model convergence. Further, it is with these values that coefficient summaries and the like can be generated. Below, we will walk through the results discussed in the article.

7 Results from Bayesian Model

7.1 Figure 4

To generate the figure, you'll have to have the numbered directories each with its own two "CODAchain" files and with its own "CODAindex" file. As mentioned above these reside in: `jags1\`. Each folder contains the files for that particular study (notice that studies are associated with the order in input file `apsr_input.csv`).

To speed up the process of reading in the CODA files, it is worth making an index file for the `phi` terms.[3] We can do this as follows:

[1] The cluster consists of 142 Nehalem 5550 nodes (1,136 hyperthreaded cores) with 24 GB of memory per node. Because of the conditional nature of the MCMC simulation, we were not able to run the models in a fully parallel environment, though lots of interesting research is being done on parallelized MCMC simulation.

[2] For size reasons, we do not provide all of the CODA files, though technically we could make them available. A compressed, zipped file containing all of the CODA files for the models in our study is around 10GB. While we are happy to release these files to interested users, they will not be hosted with all of the other replication materials for immediate download.

[3] Going through all 86 studies will still take a reasonable amount of time even with this time-saving measure. The first study we encounter takes around 2 minutes to work through the second code chunk below. Extrapolating,

```
setwd(replication_directory)

source("input_files/study_map.R")

for(i in 1:86){
    setwd(paste(replication_directory,"/jags1/",i,sep=""))
    rl <- readLines("CODAindex.txt")
    writeLines(rl[grep("phi", rl)], "phiindex.txt")
    setwd(replication_directory)
}
```

The code below calculates significance for the coefficients from each study.

```
setwd(replication_directory)

rmna <- function(x)x[-which(is.na(x))]
input <- read.csv("input_files/apsr_input.csv")[,-1]
ctls <- input[,12:ncol(input)]
ctls <- cbind(ctls[,2], ctls[,-2])
names(ctls)[1] <- "control.age"
ctls <- apply(ctls, 2, as.character)
ctl.sig <- matrix(ncol=ncol(ctls), nrow=nrow(ctls))

for(j in 1:86){
setwd(paste(replication_directory,"/jags1/",j,sep=""))
cod1 <- read.coda("CODAchain1.txt", "phiindex.txt")
cod2 <- read.coda("CODAchain2.txt", "phiindex.txt")
cphi <- rbind(cod1, cod2)

tmp <- strsplit(gsub("]", "", sapply(strsplit(colnames(cphi),
    split="[", fixed=T), function(x)x[2]), fixed=T), split=",", fixed=T)
phi.cols <- as.numeric(tmp[[length(tmp)]][1])
phi.rows <- as.numeric(tmp[[length(tmp)]][2])

cs <- t(combn(phi.cols, 2))
res <- NULL

for(i in 1:phi.rows){
cp1 <- cphi[,grep(paste(",", i,"]",  sep=""), colnames(cphi), fixed=T)]
cp1t <- lapply(1:nrow(cs), function(x)apply(cp1[,cs[x,]], 1, diff))
res <- rbind(res, sapply(cp1t, function(x)min(mean(x > 0), mean(x < 0))))
}
```

it should take around 3 hours to crunch through the code below. I have include an incremental write-out of the results. Thus, if something happens in the middle, you can see where the program broke down and start from that point to move forward without loss of the previously generated results.

```
sig <- apply(res, 1, function(x)mean(x< .025))
sig <- sig[-length(sig)]
ind <- which(input[,4] == map.mat[j,2])

dat <- read.dta(paste(replication_directory,
                          "/country_files/",
                          as.character(input[ind,1]),sep=""))
ctrl_names <- rmna(ctls[ind, ])
ctrl_names <- as.character(unlist(ctrl_names))
ctrl_names <- ctrl_names[which(ctrl_names %in% names(dat))]
ctrl_mat <- dat[,match(ctrl_names, names(dat))]
ctrl.na <- apply(ctrl_mat, 2, function(x)mean(is.na(x)))

if(max(ctrl.na) == 1){
ctrl_mat <- ctrl_mat[,-which(ctrl.na == 1)]
}

ctrl_names <- names(ctrl_mat)
ctl.sig[ind, match(ctrl_names, ctls[ind, ])] <- sig

setwd(replication_directory)
write.csv(ctl.sig, "out_files/ctl_sig.txt")
cat(j, " ", Sys.time(), "\n", file="out_files/counter.txt", append=T)
setwd(replication_directory)
}
```

After the previous step is done, you will have two files in the directory `out_files\` these are: `counter.txt` and `ctl_sig.txt`. You will also have `ctl.sig` as an object in your workspace, but these text files can be read in rather than recreating them later if desired. Now, we can make the figure. Two files will be generated in the `out_files\` directory: `figure_4_left_panel.pdf` and `figure_4_right_panel.pdf` corresponding both to figure 4 in the paper.

```
setwd(replication_directory)
ctl.list <- lapply(1:86, function(x)ctls[x,])
rmna <- function(x)x[-which(is.na(x))]
ctl.list <- lapply(ctl.list, rmna)
tab <- table(unlist(lapply(ctl.list, function(x)names(x))))
tab <- tab[order(tab)]

ctl.sig <- read.csv("out_files/ctl_sig.txt")
ctl.sig <- ctl.sig[,-1]
colnames(ctl.sig) <- colnames(ctls)

ctl.sig[, "control.occ1"] <- apply(ctl.sig[, grep("occ", colnames(ctl.sig))],
     1, max, na.rm=T)
ctl.sig[which(ctl.sig[, "control.occ1"] == "-Inf"), "control.occ1"] <- NA
ctl.sig[, "control.postmat1"] <- apply(ctl.sig[, grep("postmat",
```

```
     colnames(ctl.sig))], 1, max, na.rm=T)
ctl.sig[which(ctl.sig[, "control.postmat1"] == "-Inf"), "control.postmat1"] <- NA
ctl.sig <- ctl.sig[, -grep("postmat2", colnames(ctl.sig))]
ctl.sig <- ctl.sig[, -grep("occ2", colnames(ctl.sig))]

ctl.list <- lapply(1:86, function(x)ctl.sig[x,])
rmna <- function(x)x[-which(is.na(x))]
ctl.list <- lapply(ctl.list, rmna)
tab <- table(unlist(lapply(ctl.list, function(x)names(x))))
tab <- tab[order(tab)]

t2 <- tab[which(tab > 20)]
which(colnames(ctl.sig) %in% names(t2))

ctl.plot <- ctl.sig[,1:13]
cabbrev <- as.character(input[,2])

cabbrev[which(cabbrev == "Australia")] <- "AUL"
cabbrev[which(cabbrev == "Austria")] <- "AUT"
cabbrev[which(cabbrev == "Belgium")] <- "BEL"
cabbrev[which(cabbrev == "CzechRepublic")] <- "CZE"
cabbrev[which(cabbrev == "Denmark")] <- "DEN"
cabbrev[which(cabbrev == "Finland")] <- "FIN"
cabbrev[which(cabbrev == "France")] <- "FRA"
cabbrev[which(cabbrev == "Germany")] <- "GER"
cabbrev[which(cabbrev == "Greece")] <- "GRE"
cabbrev[which(cabbrev == "Hungary")] <- "HUN"
cabbrev[which(cabbrev == "Iceland")] <- "ICE"
cabbrev[which(cabbrev == "Ireland")] <- "IRE"
cabbrev[which(cabbrev == "Italy")] <- "ITA"
cabbrev[which(cabbrev == "Luxembourg")] <- "LUX"
cabbrev[which(cabbrev == "Malta")] <- "MAL"
cabbrev[which(cabbrev == "Netherlands")] <- "NET"
cabbrev[which(cabbrev == "NewZealand")] <- "NWZ"
cabbrev[which(cabbrev == "Norway")] <- "NOR"
cabbrev[which(cabbrev == "Portugal")] <- "POR"
cabbrev[which(cabbrev == "Romania")] <- "ROM"
cabbrev[which(cabbrev == "SlovakRepublic")] <- "SVK"
cabbrev[which(cabbrev == "Slovenia")] <- "SLV"
cabbrev[which(cabbrev == "Sweden")] <- "SWE"

abbrev <- paste(cabbrev, sapply(input[,7], function(x)substr(as.character(x),
     start=3, stop=4)), sep=" ")
xseq <- seq(from=1, to=8, length=13)
pdf("out_files/figure_4_left_panel.pdf", height=11, width=4)
par(mar=c(6,0,0,1))
```

```
plot(c(-1,8), c(1.5,43), xlab="", ylab="", type="n", axes=F, ylim=c(1.75,42))
text(rep(0.5, 43), 43:1,abbrev[1:43], pos=2, cex=.8)
r <- .2
compi <- 1
for(i in 43:1){
    for(j in 1:13){
        if(!is.na(ctl.plot[compi,j])){
        if(ctl.plot[compi,j] == 0){
            draw.circle(xseq[j],i,radius = r)
        }
        if(ctl.plot[compi,j] == 1){
            draw.circle(xseq[j],i,radius=r, col="gray65")
        }
        if(ctl.plot[compi,j] > 0 & ctl.plot[compi,j] < 1){
            floating.pie(xseq[j],i,c(10*ctl.plot[compi,j],
            10*(1-ctl.plot[compi,j])), radius=r, col=c("gray65", "white"))
        }
    }
}
compi <- compi+1
}
varnames <- c("Age", "Econ", "Sex", "Educ", "Relig", "Urban", "Income", "Union",
     "Dem Sat", "Class", "Job", "EU", "Post Mat")
axis(1, at=xseq, labels=varnames, las=2, cex.axis=.8)
xmean <- apply(cbind(xseq[1:12], xseq[2:13]), 1, mean)
abline(v=xmean, lty=2, col="gray70")
dev.off()

pdf("out_files/figure_4_right_panel.pdf", height=11, width=4)
par(mar=c(6,0,0,1))
plot(c(-1,8), c(1.5,43), xlab="", ylab="", type="n", axes=F, ylim=c(1.75,42))
text(rep(0.5, 43), 43:1,abbrev[44:86], pos=2, cex=.8)
r <- .2
compi <- 44
for(i in 43:1){
    for(j in 1:13){
        if(!is.na(ctl.plot[compi,j])){
        if(ctl.plot[compi,j] == 0){
            draw.circle(xseq[j],i,radius = r)
        }
        if(ctl.plot[compi,j] == 1){
            draw.circle(xseq[j],i,radius=r, col="gray65")
        }
        if(ctl.plot[compi,j] > 0 & ctl.plot[compi,j] < 1){
            floating.pie(xseq[j],i,c(10*ctl.plot[compi,j],
            10*(1-ctl.plot[compi,j])), radius=r, col=c("gray65", "white"))
```

```
        }
    }
}
compi <- compi+1
}
varnames <- c("Age", "Econ", "Sex", "Educ", "Relig", "Urban", "Income", "Union",
     "Dem Sat", "Class", "Job", "EU", "Post Mat")
axis(1, at=xseq, labels=varnames, las=2, cex.axis=.8)
xmean <- apply(cbind(xseq[1:12], xseq[2:13]), 1, mean)
abline(v=xmean, lty=2, col="gray70")
dev.off()
```

7.2 Figures 5 and 6

Again, it is useful to take some time-saving measures here. We have already made our lives easy by making the first monitored parameter (i.e., the first 1250 values in every "CODAchain" file) β and the second monitored parameter λ. So, we only have to read in the first 2500 lines of each chain file, which will save a considerable amount of time. To make this easiest, using the `coda` package, we need to make subsets of the chain and index files as follows:

```
setwd(replication_directory)

for(i in 1:86){
    setwd(paste(replication_directory,"/jags1/",i,sep=""))
    in1 <- readLines("CODAchain1.txt", n=2500)
    in2 <- readLines("CODAchain2.txt", n=2500)
    ind <- readLines("CODAindex.txt", n=2)
    writeLines(in1, con="lbchain1.txt")
    writeLines(in2, con="lbchain2.txt")
    writeLines(ind, con="lbindex.txt")
    setwd(replication_directory)
    print(i)
}


setwd(replication_directory)
source("input_files/study_map.R")
input <- read.csv("input_files/apsr_input.csv")[,-1]

myres <- list()

for(i in 1:86){
setwd(paste(replication_directory,"/jags1/",i, sep=""))
    myres[[i]] <- rbind(read.coda("lbchain1.txt","lbindex.txt"),
read.coda("lbchain2.txt","lbindex.txt"))
setwd(replication_directory)
}
```

```
setwd(replication_directory)
input <- input[match(map.mat[,2], input[,4]), ]
cnames <- read.table("input_files/country_map.txt", header=T)
blres <- t(sapply(myres, function(x)apply(x, 2, quantile, probs=c(.5,.025,.975))))
input.ctry <- as.character(cnames[match(input[,3], cnames[,2]), 1])
input.date <- as.Date(paste("01",input[,6], input[,7], sep="-"), "%d-%m-%Y")

pdf("out_files/figure_5.pdf", height=7, width=7)
trellis.par.set(strip.background = list(col="gray90"))
print(
xyplot(blres[,1] ~ input.date | input.ctry, ylim = c(0,1.1), pch=16, cex=.4,
     col="black", as.table=T,  par.strip.text=list(cex=.7),
     xlab="Time", ylab=expression(beta), scales = list(x = list(rot = 45)),
     panel=function(x,y,subscripts){
        panel.segments(x,blres[subscripts,2], x,blres[subscripts,3])
        panel.points(x,y,pch=16, cex=.4, col="black")
    }
    )
)
dev.off()

pdf("out_files/figure_6.pdf", height=7, width=7)
trellis.par.set(strip.background = list(col="gray90"))
print(
xyplot(blres[,4] ~ input.date | input.ctry, ylim=c(min(blres[,5]), max(blres[,6])),
     pch=16, cex=.4, col="black", as.table=T,  par.strip.text=list(cex=.7),
     xlab="Time", ylab=expression(lambda), scales = list(x = list(rot = 45)),
     panel=function(x,y,subscripts){
        panel.segments(x,blres[subscripts,5], x,blres[subscripts,6])
        panel.points(x,y,pch=16, cex=.4, col="black")
        panel.abline(h=0, col="gray60", lty=2)
    }
    )
)
dev.off()
```

8 Sensitivity Testing

We did a number of sensitivity tests, the most extensive of which requires estimating the equivalent Bayesian models in the more standard conditional logit framework. This can be done very easily in JAGS by replacing the model run for the main results in the article, with the one below[4]:

```
data{
```

[4]You'll notice that some of the important parameters, like β are still being simulated in the model, but without being confronted with data, so they do little to the model. If anything, they induce a bit of inefficiency, but not enough to warrant the time it would take to re-write and re-run the data and model generating process.

```
for(m in 1:Nctrl){
    phi[Np,m] <- 0
}
}
model{
for(i in 1:N){
lr[i] ~ dnorm(0,1)
ctrlmat[i,1] ~ dnorm(50,.1)
for (k in 2:(Nctrl-1)) {
     ctrlmat[i,k] ~ dcat(prob.mat[k, ])
}
vote[i] ~ dcat(p[i,])
    for (k in 1:Np) {
            sindist[i,k] <- pow(lr[i] - plmat[k,1], 2)
        log(q[i,k]) <- lambda*sindist[i,k] + inprod(ctrlmat[i, ], phi[k,])
        p[i,k] <- q[i,k]/(sum(q[i,]))
        }
}
for (m in 1:Nctrl) {
    for(l in 1:(Np-1)){
    phi[l,m] ~ dnorm(0, .01)
    }
}
for (i in 1:N) {
    beta[i] ~ dbern(prob.beta)
}
prob.beta ~ dunif(0,1)
lambda ~ dnorm(0,.1)
}
```

Just like above, I place the results in the directory called `jags2`. Once the models have been run in a similar fashion to the ones above (in this case by running `jags mnl3.cmd`), you can cake the predicted probabilities (monitored for each case in the CODA files) from both models and calculate the predicted probabilities. Going through the for loop below one at a time took my computer around 8 hours. To ensure minimal loss in the case that your computer crashes, we include incremental write-out statements at each iteration of the loop to save the necessary information. Thus, in each directory, you will have a file called "pre.csv" which has the essential information to calculate whatever statistics you want relating to the two models.

```
setwd(replication_directory)

source("input_files/study_map.R")
for(m in 1:86){
    setwd(paste(replication_directory,"/jags2/",m, sep=""))
    cod1 <- rbind(read.coda("CODAchain1.txt","CODAindex.txt"),
        read.coda("CODAchain2.txt", "CODAindex.txt"))
```

```
probs <- cod1[, grep("p[", colnames(cod1), fixed=T)]
numi <- max(as.numeric(sapply(strsplit(sapply(strsplit(colnames(probs),
    split="[", fixed=T), function(x)x[2]), split=",", fixed=T), function(x)x[1])))
nump <- max(as.numeric(gsub("]", "", sapply(strsplit(sapply(strsplit(
    colnames(probs), split="[", fixed=T), function(x)x[2]), split=",", fixed=T),
    function(x)x[2]), fixed=T)))

prob.array <- array(c(t(probs)), dim=c(numi,nump,2500))
votes <- t(cod1[, grep("vote", colnames(cod1), fixed=T)])

tab <- t(apply(votes, 2, table))
ptab <- prop.table(tab, 1)
epmc <- diag(tab %*% t(ptab))

test <- sapply(1:2500, function(x)prob.array[cbind(1:numi, votes[,x], x)])

setwd(paste(replication_directory, "/jags1/",m, sep=""))
cod1a <- rbind(read.coda("CODAchain1.txt","CODAindex.txt"),
    read.coda("CODAchain2.txt", "CODAindex.txt"))

probsa <- cod1a[, grep("p[", colnames(cod1a), fixed=T)]
prob.arraya <- array(c(t(probsa)), dim=c(numi,nump,2500))
votesa <- t(cod1a[, grep("vote", colnames(cod1a), fixed=T)])
taba <- t(apply(votesa, 2, table))
ptaba <- prop.table(taba, 1)
epmca <- diag(taba %*% t(ptaba))

testa <- sapply(1:2500, function(x)prob.arraya[cbind(1:numi, votesa[,x], x)])

t1 <- apply(prob.array, c(1,3), which.max)
sums <- sapply(1:2500, function(x)sum(votes[,x] == t1[,x]))

t1a <- apply(prob.arraya, c(1,3), which.max)
sumsa <- sapply(1:2500, function(x)sum(votesa[,x] == t1a[,x]))

null <- apply(apply(votes, 2, table), 2, max)
nulla <- apply(apply(votesa, 2, table), 2, max)

write.csv(data.frame(numcp_cl = sums, null_cl = null,
    epcp_cl = apply(test, 2, sum), epmc_cl = epmc,
    numcp_us = sumsa, null_us = nulla, epcp_us = apply(testa, 2,sum),
    epmc_us = epmca), paste(replication_directory, "/jags2/",m,
    "/pre.csv", sep=""))

cat(m, as.character(Sys.time()), "\n",
    file="precounter.txt", append=T)
```

```
}
```

The code below uses the files made at each iteration of the preceding loop.

```
res <- NULL
setwd(replication_directory)
for(i in 1:86){
    source(paste("jags2/",i, "/mnldat.txt", sep=""))
    dat <- read.csv(paste("jags2/",i, "/pre.csv", sep=""))
    epre <- quantile((dat$epcp_us - dat$epcp_cl)/(length(vote)-dat$epcp_cl),
        probs=c(.05, .5))
    pre <- quantile((dat$numcp_us - dat$numcp_cl)/(length(vote) - dat$numcp_cl),
        probs=c(.05, .5))
    res <- rbind(res, c(map.mat[i,2], pre, epre))
}
```

You can see how many times our model beats the conventional conditional logit model by doing:

```
apply(res[,c(2,4)], 2, function(x)c(sum(x > 0), mean(x > 0)))
```

The following code generates figure 7 and saves it as a pdf file in the `out_files` folder.

```
d1 <- density(res[,3])
d2 <- density(res[,5])
d1$y <- d1$y/max(d1$y)
d2$y <- d2$y/max(d2$y)

pdf("out_files/figure_7.pdf", height=4, width=4)
plot(d1, xlab="Proportional Reduction in Error", ylab="", axes=F, main="")
lines(d2, lty=2)
axis(1)
box()
legend("topright", c("PRE", "ePRE"), lty=c(1,2), inset=.01)
mtext("Density", side=2, line=1)
dev.off()
```

8.1 Significance of Controls

Using the previously generated `ctl.sig`, you can calculate the percentage of models in which coefficients are significant (for at least one comparison) as follows:

```
apply(ctl.sig[,1:13], 2, function(x)mean(x > 0, na.rm=T))
```